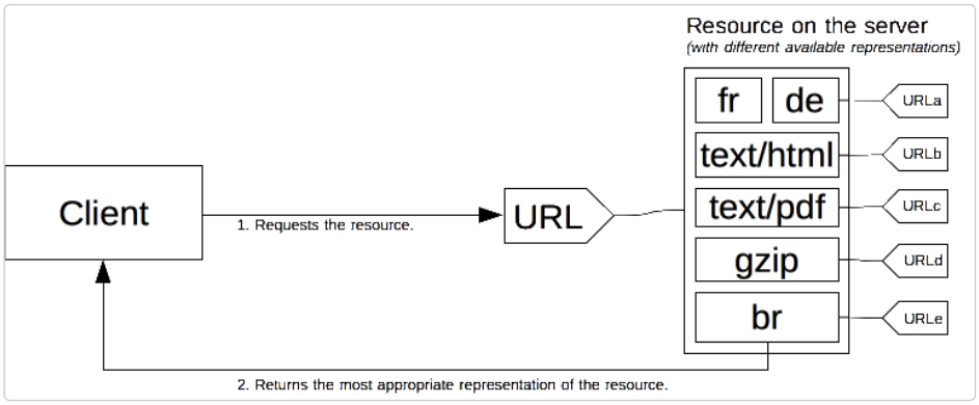
Negociação de conteúdo

No HTTP, **negociação de conteúdo** é o mecanismo utilizado para servir diferentes representações de um recurso para a mesma URI, ajudando o agente de usuário a especificar qual representação é mais adequada para o usuário (por exemplo, qual idioma do documento, qual formato de imagem ou qual codificação de conteúdo).

Princípios da negociação de conteúdo

Um documento específico é chamado de recurso. Quando um cliente deseja obter um recurso, ele o solicita por meio de uma URL.
O servidor usa essa URL para escolher uma das variantes disponíveis — cada variante é chamada de **representação** — e retorna uma representação específica ao cliente.
O recurso geral, assim como cada uma das representações, possui uma URL específica.
A negociação de conteúdo determina **como uma representação específica é escolhida** quando o recurso é solicitado.
Existem várias formas de negociação entre o cliente e o servidor.



Nota: Você encontrará algumas desvantagens da negociação de conteúdo HTTP em uma página wiki da WHATWG.
O HTML fornece alternativas à negociação de conteúdo via, por exemplo, o elemento <source>.

A representação mais adequada é identificada por um dos dois mecanismos:

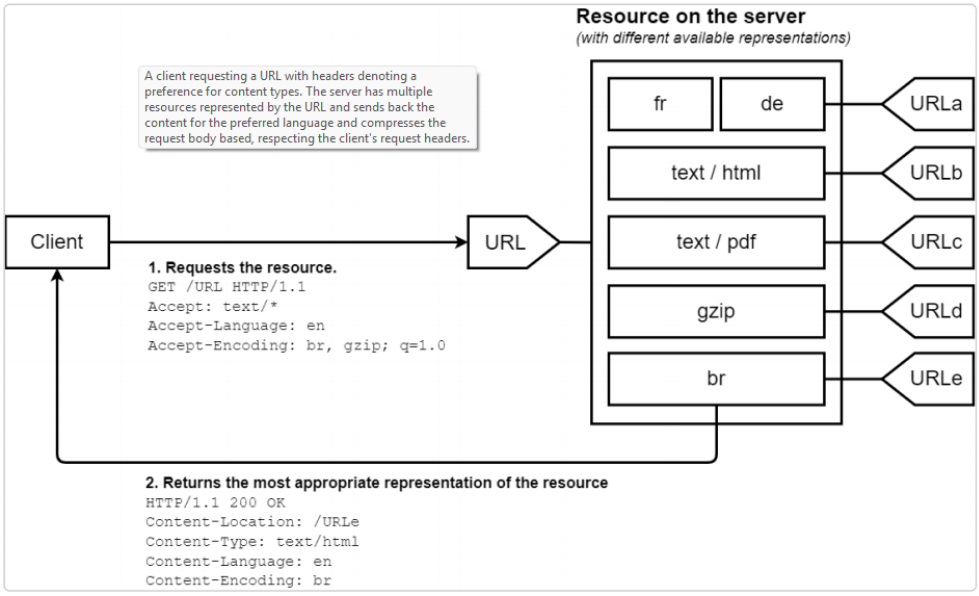
- **Cabeçalhos HTTP específicos enviados pelo cliente**
  (negociação orientada pelo servidor ou negociação proativa),
  que é a forma padrão de negociar um tipo específico de recurso.
- **Códigos de resposta HTTP 300 (Multiple Choices), 406 (Not Acceptable), 415 (Unsupported Media Type)** enviados pelo servidor
  (negociação orientada pelo agente ou negociação reativa), usados como mecanismos de fallback.

Com o tempo, outras propostas de negociação de conteúdo, como a **negociação de conteúdo transparente** e o cabeçalho Alternates, foram sugeridas.
No entanto, elas não ganharam tração e foram abandonadas.

Negociação de conteúdo orientada pelo servidor

Na **negociação orientada pelo servidor**, ou **negociação proativa**, o navegador (ou qualquer outro tipo de agente de usuário) envia vários cabeçalhos HTTP junto com a URL.
Esses cabeçalhos descrevem a preferência do usuário.
O servidor os utiliza como sugestões, e um algoritmo interno escolhe o melhor conteúdo a ser servido ao cliente.
Se não conseguir fornecer um recurso adequado, poderá responder com 406 Not Acceptable ou 415 Unsupported Media Type e definir cabeçalhos para os tipos de mídia que suporta

(por exemplo, usando Accept-Post ou Accept-Patch para requisições POST e PATCH, respectivamente).
O algoritmo é específico do servidor e **não é definido pelo padrão**.
Veja o algoritmo de negociação do Apache.



O padrão HTTP/1.1 define uma lista de cabeçalhos padrão que iniciam a negociação orientada pelo servidor (como Accept, Accept-Encoding e Accept-Language).
Embora o User-Agent não esteja nesta lista, às vezes ele também é usado para enviar uma representação específica do recurso solicitado.
No entanto, isso nem sempre é considerado uma boa prática.
O servidor usa o cabeçalho Vary para indicar **quais cabeçalhos** foram efetivamente usados na negociação de conteúdo (mais precisamente, os cabeçalhos da requisição associados), de modo que os caches possam funcionar de forma ideal.

Além disso, há uma proposta experimental para adicionar mais cabeçalhos à lista de disponíveis, chamada **client hints**.
Client hints anunciam **qual tipo de dispositivo** o agente de usuário está utilizando (por exemplo, computador desktop ou dispositivo móvel).

Mesmo que a negociação de conteúdo orientada pelo servidor seja a forma mais comum de se acordar sobre uma representação específica de um recurso, ela possui várias desvantagens:

- O servidor **não tem conhecimento total sobre o navegador**.
  Mesmo com a extensão Client Hints, ele não possui uma compreensão completa das capacidades do navegador.
  Diferente da negociação reativa, onde o cliente faz a escolha, a escolha feita pelo servidor é sempre, de certa forma, arbitrária.
- As informações do cliente são bastante verbosas (a compressão de cabeçalhos no HTTP/2 mitiga esse problema) e representam um **risco à privacidade** (por exemplo, *HTTP fingerprinting*).

Cabeçalho Accept

O cabeçalho Accept permite ao cliente indicar **quais tipos MIME** ele pode processar.
Esse cabeçalho pode ser usado para solicitar um **documento em HTML**, **em texto simples**, **em JSON** ou em qualquer outro formato apropriado.
O servidor escolhe um desses formatos e o envia de volta, utilizando o mesmo tipo MIME no cabeçalho Content-Type.

```bash
Accept: text/html, application/xhtml+xml, application/xml;q=0.9, */*;q=0.8
```

O cliente também pode atribuir **níveis de prioridade** a cada tipo MIME por meio de um **peso de qualidade (quality value)**, usando o parâmetro q.
Esses pesos vão de 0 (não aceitável) até 1 (aceitável com máxima prioridade).
Por exemplo, com o seguinte cabeçalho:

```bash
Accept: text/html, application/xhtml+xml, application/xml;q=0.9, */*;q=0.8
```

- text/html e application/xhtml+xml têm prioridade total (q=1, valor padrão)
- application/xml é um pouco menos preferido (q=0.9)
- */* representa qualquer tipo de conteúdo, aceito com baixa prioridade (q=0.8)

Se o servidor não conseguir fornecer nenhuma das opções listadas, ele pode responder com um código de status **406 Not Acceptable**.

Cabeçalho Accept-Charset

O cabeçalho Accept-Charset permite ao cliente indicar **quais conjuntos de caracteres** ele pode entender.

```makefile
Accept-Charset: utf-8, iso-8859-1;q=0.5
```

Esse cabeçalho raramente é usado hoje em dia, pois a maioria dos clientes e servidores **usa UTF-8 por padrão**.
O HTTP/1.1 afirma que, na ausência desse cabeçalho, o padrão deve ser **iso-8859-1**, mas na prática, essa regra foi ignorada em favor do UTF-8.

Cabeçalho Accept-Encoding

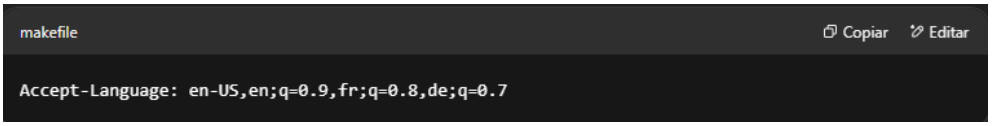
O cabeçalho Accept-Encoding permite ao cliente indicar **quais algoritmos de compressão** ele suporta.
O servidor usa essa informação para enviar de volta o conteúdo compactado com o método mais apropriado (por exemplo, gzip, br, deflate).

```makefile
Accept-Encoding: gzip, compress, br
```

Esse cabeçalho é amplamente usado para **otimizar o tempo de resposta** e a **largura de banda consumida**.
Assim como nos demais casos, o servidor deve indicar o tipo de codificação de conteúdo utilizada na resposta, por meio do cabeçalho Content-Encoding.

Cabeçalho Accept-Language

O cabeçalho Accept-Language permite ao cliente indicar **quais idiomas são aceitáveis** na resposta.



Neste exemplo:

- en-US é a variante americana do inglês, com prioridade máxima
- en é o inglês em geral, um pouco menos prioritário
- fr (francês) e de (alemão) são menos preferidos

Esse cabeçalho é usado com frequência para **internacionalização** de páginas web.
No entanto, ele **não substitui** métodos de personalização mais robustos, como seleção de idioma via configuração do usuário, cookies ou parâmetros de URL.
O idioma escolhido deve ser refletido no cabeçalho de resposta Content-Language.

Cabeçalho User-Agent

Embora não faça parte do conjunto de cabeçalhos padrão para negociação de conteúdo, o User-Agent é **frequentemente usado por servidores** para adaptar a resposta conforme o tipo de navegador.
Por exemplo, pode-se servir uma página simplificada para navegadores antigos, ou personalizar o conteúdo para navegadores móveis.
Esse uso é comum, mas **controverso**, pois o User-Agent **é de difícil padronização**, e sua análise pode gerar erros.
Além disso, é um vetor conhecido para **técnicas de fingerprinting**, comprometendo a privacidade.

Negociação de conteúdo orientada pelo agente

Na negociação **orientada pelo agente**, ou **negociação reativa**, o servidor fornece uma lista de representações disponíveis, e **o cliente escolhe** qual delas é a mais apropriada.
Ela é implementada pelo servidor retornando o código de status **300 Multiple Choices**, junto com uma lista de URLs apontando para as diferentes variantes disponíveis.

Este mecanismo é raramente usado, pois **muitos navegadores não conseguem processar adequadamente** a resposta 300 Multiple Choices.
Em vez disso, o uso de **redirecionamento baseado em JavaScript** ou elementos HTML como <source> são preferidos.
Por exemplo, o elemento <picture> permite aos navegadores escolher qual imagem carregar com base nas condições do lado do cliente (como resolução de tela).

Negociação de conteúdo baseada em URI

A negociação de conteúdo baseada em URI consiste em **associar diferentes representações a diferentes URLs**.
Isso elimina a negociação automática: o servidor não escolhe qual representação enviar, o cliente é **responsável por escolher explicitamente** a URL correta.

Por exemplo, considere uma imagem chamada logo disponível em três formatos: SVG, PNG e WebP.
Em vez de servir a imagem via https://exemplo.com/logo e usar negociação automática, o servidor pode usar:

- https://exemplo.com/logo.svg
- https://exemplo.com/logo.png
- https://exemplo.com/logo.webp

Isso simplifica o cache, torna a resposta mais previsível, e **evita ambiguidade** na URL.
É também o método preferido pelo WHATWG (Web Hypertext Application Technology Working Group), e o mais usado na prática para internacionalização de páginas.

Cabeçalhos de apoio à negociação de conteúdo

Os seguintes cabeçalhos são usados **pelo servidor** para indicar quais mecanismos de negociação foram utilizados, ou para facilitar o processo de seleção por parte do cliente:

- **Vary**: Indica quais cabeçalhos foram usados para selecionar a resposta.
  Isso permite que caches intermediários armazenem corretamente as variações do conteúdo.
- **Content-Language**: Indica o idioma da resposta.
- **Content-Encoding**: Indica o tipo de compressão usado na resposta.
- **Content-Type**: Indica o tipo MIME da resposta.
- **Alternates** (proposta abandonada): Indica outras representações possíveis de um recurso.
  Era usada com a resposta 406 Not Acceptable, mas **não teve adoção prática significativa**.

Resumo

A negociação de conteúdo é um mecanismo poderoso do HTTP, permitindo que um único recurso tenha múltiplas representações e que o cliente ou o servidor escolha a mais apropriada.
Apesar de útil, ela é **pouco usada** diretamente na prática devido a limitações de interoperabilidade e problemas de cache.

As abordagens modernas tendem a:

- **Evitar negociação automática**,
- Preferir **URLs específicas para cada representação**,
- Usar elementos HTML e técnicas do lado do cliente para escolher a melhor variante (como <picture>, <source>, ou JavaScript).